

SUMMARY OF THE MPEC'05 WORKSHOP DISCUSSION

After the keynote talk and the ten presentations that took place during the workshop, we identified some topics of discussion which served to raise questions and give opinions. Because there were many topics, we had to skip some of them. The skipped topics are enumerated below and were not developed.

- **Quality requirements and models**

- Is there a chance to reach a consensus and obtain a unique quality model? There exists ISO9126 and many others, and it is impractical to discard them.
- How to evaluate OTS products which have so little documentation and which are not easy to verify? How do you know what the quality characteristics of an OTS component are, how to verify that the OTS products have the required level and how to know if they are good enough for your use? How to bridge the gap between the quality stated for the OTS products and the quality required for the whole system?
- Can (does it pay) technical quality models be enlarged with other type of non-technical characteristics? Problem: most of these characteristics will not be of the product but rather of the vendor.
- Are quality models stable (specially ISO)?
- Problems: ISO's quality hierarchy is very flat. In the general case, we need even graphs (labelled –NFR). This is done also in SEI architecting methods (ATAM...).
- Proliferation of standards... What should we do if two OTS products are described using different standards? There was not an optimistic view about quality model unification in the workshop audience.
- Is there a method for evaluating –ilities? What about the cost? We have methods, but they may have some scalability problems. Problem: how do we assess a OTS product before we buy it and we put it into the system? There seems to be no one-size-fits-all method.

- **General framework for OTS issues**

- Is the OTS community generating confusion (e.g., by given different names to the same thing)? Some comparison between current existing approaches could be useful.
- Is it feasible to have a unique framework (e.g., selection) of OTS-related method components? The aim is to provide general concepts that are easy to tailor in particular environments (a kind of standard with definitions of tasks, goals, roles) and still leave flexibility to make detailed definitions.
- Distinguishing very generic and more specific concepts was discussed.
- Commitment was reached between several of the participants to search for consensus and openness with regard to reusable OTS-related method components.

- **Repositories**

- How to manage extreme evolution in repositories of OTS components? How can you trust (and even find) information given by others? Some discouraging past experiences were discussed... There are even more radical views (keynote at CBSE).

- Factors affecting: How quickly components evolve and what are their expected lifetimes? These factors may be domain-dependant. Also, the amount of information you aim to include as specification of the components may be critical.
- More interesting than the repository itself, may be to have a structure that guides the process of finding the components of interest. We need to find information before we actually pay for the components. How should valid information be put in the repository and kept updated? Is it possible to act as in the Web Services community with the UDDIs? Who puts the information in the repository? Perhaps money helps ;-). An existing trend is just to gather information without prototyping, as a kind of normalization process. It is important to remind the eCots philosophy (open community).
- **Empirical analysis**
 - Why there is a gap proposals-reality? First, they do not know for sure if they will have adequate ROI. An experience was reported where there was a risk assessment phase and results were not clear to adopt these methods.
 - Putting OTS-related concepts in the context general of SE can help to show that it is not a brand new world.
 - Major problems are a lack of skills as well as resources and budget to make proper OTS evaluation. Even if there will be ROI, immediate milestone do not allow careful evaluation. There is a tendency to reuse COTS that has already been selected without adequate evaluation. Obtaining adequate training time is also a problem.
- **Tool presentations**
- **Techniques (MCDM, ...)**
- **Business models**
 - Vendors put money in consultants. Can we expect this in our context?
 - Government: may hire an independent reviewer to assess the process of making OTS decisions (it is a role to consider).
- **Phases of processes: selection, testing**
 - What are the differences between general architecting and OTS architecting? Renegotiation of the architecture may be important. Interoperability problems often exist in an OTS-based solution. Problems may emerge when you are integrating OTS products into the rest of the architecture. And do not forget evolution problems.
- **Requirements and architectures**
 - If we do not have good requirements (especially quality requirements), how can we properly evaluate the architecture? Evaluation against perfection or against expectations? Key factor: iteration and intertwining (negotiation).
 - It is important to know what is not negotiable.
 - Meaning of modularization in architecting OTS systems.
- **Specification of OTS components**
- **Evolution**